## Python y tipos base

### TUPLA

```
a = (3, 5)
a[0]                    : 3
a[0] = 1                : ERROR (inmutable)
t1 = (0,)               # tupla de 1 elemento.
```

### LISTA

```
b = [5, 7]
b[0]                    : 5
b[1] = 9                : [5, 9]
b.append( 3)            : [5, 9, 3]
b.insert( 0, 2)         : [2, 5, 9, 3]
b.remove( 9)            : [2, 5, 3]

del b[1]                : [2, 3]
b.len()                 : ERROR
len(b)                  : 2

c = [[1, 2], [3, 4]]  # Lista de listas, no mat.
C[0,0]                  : ERROR
c[0][0]                 : 1

a = range(5)            : range(0, 5)
list(a)                 : [0, 1, 2, 3, 4]
b = range(3,10,2)       : [3, 5, 7, 9]
c = [ x*2+1 for x in range(1,5) ]
                        : [3, 5, 7, 9]
xrange?
```

### Slicing

```
b[1:3]                  : [5, 7]
b[:2]                   : [3, 5]
b[3:]                   : [9]
b[-2:]                  : [7, 9]
b[::2]                  : [3, 7]
c[1:3] = [1, 2]         : [3, 1, 2, 9]
```

### NumPy

```
import numpy as num   # np (ver: import this)

u = num.array([5,7])  : array([ 5, 7])
u = num.zeros(2)      : array([ 0., 0.])
v = num.zeros((1, 3)) : array([[ 0., 0., 0.,]])
m = num.ones((3,2))   : array([[ 1., 1.],
                               [ 1., 1.],
                               [ 1., 1.]])

u.size                : 2
v.size                : 3
m.size                : 6

u.shape               : (2,)
v.shape               : (1, 3)
m.shape               : (3, 2)

u[0]                  : 0.0
v[0]                  : array([ 0., 0., 0.])
v[0][0]               : 0.0
v[0,0]                : 0.0

v[:] = [2, 3, 4]      : array([[ 2., 3., 4.]])
u[:] = 2              : array([ 2., 2.])
v[:] = 3              : array([[ 3., 3., 3.]])

num.dot( m, u)
m.dot( u)             : array([ 4., 4., 4.])

num.dot( v, m)
v.dot( m)             : array([[ 9., 9.]])

m*u
u*m                   : array([[ 2., 2.],
                               [ 2., 2.],
                               [ 2., 2.]])
m*v                   : ERROR

m*v.T                 : array([[ 3., 3.],
                               [ 3., 3.],
                               [ 3., 3.]])

u.T.shape == u.shape  : True
v.T.shape == v.shape  : False

u.reshape((2,1))      : array([[ 2.],
                               [ 2.]])


Ver: inner, outer, prod, multiply... flatten()
Ver: num.resize vs {obj}.resize
```

### Condicionales

```
y = num.array([ -1, 0, 1])

num.where( y>0, 1, 0) : [ 0, 0, 1]
(y>0)                 : [False, False, True]
(y>0)*1               : [ 0, 0, 1]

num.[any|all](y>0)    : True|False
```

### Distribuciones aleatorias

```
x = num.random.uniform( -1, 1, (1,6))
w = num.random.normal( 0, 1, (6,3))
y = num.zeros((1,3))
y[:] = num.dot( x, w)

Ver: sum, mean, std | min, max | argmin, argmax
num.random. choice | sample | shuffle |
permutation
```

### Gráfico de funciones

```
from matplotlib import pyplot as mpl

x = num.linspace( 0, 2*num.pi, 100)
y = num.sin( x)

g = mpl.plot( x, y)
mpl.show()

Ver: num.arange

_______

m = num.random.uniform( -10, 5, (6,3))
mpl.matshow( m)
mpl.show()

Ver: ion(), ioff()
```